

Final Project Report (Team - 738184)

Project Title: Eye Disease Detection Using Deep Learning

1. Introduction

1.1 Project Overview

The project on eye disease detection using deep learning involves the utilization of a dataset sourced from Kaggle, comprising images categorized into four distinct classes: cataract, diabetic retinopathy, glaucoma, and normal. To prepare the data for model training, preprocessing steps such as normalization, data augmentation, and resizing were implemented. Four deep learning models were built and evaluated: Convolutional Neural Network (CNN), Xception, Inception V3, and VGG19. Among these, the Convolutional Neural Network (CNN) model exhibited the highest accuracy, achieving 85%, and was consequently selected for deployment. The deployment phase involved creating a web application using Flask within the Spyder environment, providing users with a straightforward interface to upload eye images for disease prediction.

1.2 Objectives

The main objective of this project is to provide a practical tool for automated eye disease detection, aiding healthcare professionals in early diagnosis and treatment planning.

Objectives of this project involves the following:

1. Download an eye disease dataset that consists of eye disease categories such as Cataract, Diabetic Retinopathy, Glaucoma and Normal.
2. Applying preprocessing steps like normalizations, data augmentation and image resizing.
3. Building deep learning algorithms such as Convolutional Neural Network (CNN), Xception, Inception V3, and VGG19 and comparing its accuracy and selecting the model that has good accuracy and minimum loss.
4. Building a web application using Flask in a spyder environment, to enable users a platform to upload eye images and predict diseases.
5. Demonstrate the potential of deep learning in medical image analysis and contribute to the development of automated diagnostic tools for eye diseases.

2. Project Initialization and Planning Phase

2.1 Define Problem Statement

Detecting several eye diseases in their early stages can be a challenging task especially when the symptoms are subtle or non-specific. Delayed detection and diagnosis can impact treatment outcomes. Some advanced diagnostic tools and techniques for eye diseases require specialized equipment and expertise that might be difficult to acquire. The current diagnosis of eye diseases faces significant challenges, impacting patient care and outcomes. Patients with conditions like age-related degeneration and diabetes struggle to receive timely and accurate diagnoses, leading to potential complications. Traditional methods often fall short in classifying diseases accurately, delaying treatment. This project aims to develop a user-friendly solution for accurate disease classification.

2.2 Project Proposal (Proposed Solution)

Below are the following proposed solutions for the above stated problem statement:

1. Development of a robust and accurate deep learning model capable of detecting and classifying cataract, diabetic retinopathy, glaucoma, and normal eye conditions.
2. Deployment of a user-friendly web application that allows healthcare professionals to upload eye images and obtain instant disease predictions.
3. Improved accessibility to automated diagnostic tools for early detection of eye diseases, leading to timely interventions and improved patient care.

Approach Used:

We'll use deep learning techniques, where the computer learns to recognize patterns in images through training on a large dataset of eye disease images. This trained model will then be able to analyze new images and identify any signs of disease.

Key Features:

Accurate detection of various eye diseases, Fast analysis of eye images, User-friendly interface for easy use by healthcare professionals and Potential for early disease detection, leading to better treatment outcomes.

2.3 Initial Project Planning

Sprint	Functional Requirement (Epic)	User Story Number	Task	Story Points	Priority	Team Members	Sprint Start Date	Sprint End Date (Planned)
Sprint-1	Project Initialization and Planning Phase	DLP-3	Define Problem Statements (Template 1)	1	Medium	Sapna Girme	16-04-2024	17-04-2024
Sprint-1	Project Initialization and Planning Phase	DLP-4	Project Proposal (Template 2)	1	Medium	Arya Talekar	16-04-2024	17-04-2024
Sprint-1	Project Initialization and Planning Phase	DLP-5	Initial Project Planning Report (Template 3)	2	Medium	Jomy Mathew	16-04-2024	17-04-2024
Sprint-2	Data Collection	DLP-12	Loading the dataset	2	Medium	Sapna Girme	17-04-2024	17-04-2024
Sprint-2	Data Collection	DLP-17	Create Training and Testing Dataset	2	High	Jomy Mathew	17-04-2024	17-04-2024
Sprint-2	Data Collection	DLP-18	Image PreProcessing	2	High	Arya Talekar	17-04-2024	17-04-2024
Sprint-3	Image Preprocessing	DLP-13	Importing the libraries	3	Medium	Arya Talekar	18-04-2024	18-04-2024
Sprint-3	Image Preprocessing	DLP-14	Configure ImageData Generator Class	3	High	Jomy Mathew	18-04-2024	18-04-2024
Sprint-3	Image Preprocessing	DLP-15	Apply ImageData Generator Functionalities	2	High	Sapna Girme	18-04-2024	18-04-2024
Sprint-4	Model Building	DLP-20	Pre-trained CNN model	3	High	Jomy Mathew	19-04-2024	20-04-2024

Sprint	Functional Requirement (Epic)	User Story Number	Task	Story Points	Priority	Team Members	Sprint Start Date	Sprint End Date (Planned)
			as feature extraction					
Sprint-4	Model Building	DLP-21	Adding Dense layers	3	High	Sapna Girme	19-04-2024	20-04-2024
Sprint-4	Model Building	DLP-22	Configure the learning process	3	High	Arya Talekar	19-04-2024	20-04-2024
Sprint-4	Model Building	DLP-23	Train the model	2	High	Jomy Mathew	19-04-2024	20-04-2024
Sprint-5	Save the model	DLP-24	Save the Model	2	Medium	Sapna Girme	20-04-2024	20-04-2024
Sprint-6	Application Building	DLP-26	Building HTML pages	2	High	Jomy Mathew	21-04-2024	22-04-2024
Sprint-6	Application Building	DLP-27	Build Python Code	2	High	Sapna Girme	21-04-2024	22-04-2024
Sprint-6	Application Building	DLP-28	Run the application	2	High	Arya Talekar	21-04-2024	22-04-2024
Sprint-7	Output	DLP-29	Output	2	Medium	Arya Talekar	21-04-2024	22-04-2024

3. Data Collection and Preprocessing

3.1 Data Collection Plan and Raw Data Sources Identified

Dataset for this project is collected from Kaggle which provides various medical imaging datasets related to eye diseases. Kaggle provides access to a variety of medical imaging datasets, including those related to ophthalmology and eye diseases. The downloaded dataset contains 4 sub-folders each with 4 eye diseases categories like cataract, glaucoma, diabetic_rethinopathy, and normal. It contains a total of 4217 images, providing a diverse collection of eye images for model training and evaluation.

Dataset Used:

Kaggle (<https://www.kaggle.com/datasets/gunavenkatdoddi/eye-diseases-classification/data>)

Dataset Characteristics:

Number of Images: 4217 images in total across all disease categories.

Distribution of Classes: Cataract: 1038 Images, Glaucoma: 1007 Images, Diabetic Retinopathy: 1098 Images, Normal: 1074 Images

Image Format: Images are likely in standard formats such as JPG, JPEG, PNG.

3.2 Data Quality Report

Data Source	Data Quality Issue	Severity	Resolution Plan
Kaggle	All the images are not of the same size.	Low	We have resized the all images into one size
Kaggle	limited diversity and quantity	Moderate	We performed data augmentation techniques including rescale, shearing, zooming, flipping, etc.
Kaggle	All the pixel values are not in the same range.	Moderate	We have scaled the pixel values of images to a standard range.

3.3 Data Preprocessing

In preprocessing the dataset for our eye disease detection project, we employed several key techniques to ensure uniformity and enhance model performance. Initially, we addressed the variation in image sizes across disease categories by resizing all images to a consistent size of 256x256 pixels. Subsequently, to prepare the resized images for model training, we applied normalization and rescaling using the ImageDataGenerator library from TensorFlow. Normalization involves adjusting pixel values to a standardized scale, normally ranging from 0 to 1. Furthermore, we leveraged data augmentation techniques, such as shearing, zooming, and flipping, also implemented using the ImageDataGenerator library.

Resizing	Each disease folder had images that were in different sizes. We have resized all the images into a size of 256x256.
Normalization	Normalized and rescaled all the images using ImageDataGenerator library from tensorflow.
Data Augmentation	Performed techniques like shearing, zooming and flipping using ImageDataGenerator library from tensorflow.

4. Model Development Phase

4.1 Model Selection Report

Model	Description
CNN	The CNN model for Eye Disease Detection employs deep learning techniques to classify eye diseases from retinal scans or fundus images. Trained on labeled datasets, it learns to extract relevant features from images, enabling accurate disease prediction. This model aids in early diagnosis, facilitating timely treatment and potentially improving patient outcomes.
VGG19	Trained on large datasets of retinal or fundus images labeled with various eye diseases, VGG19 learns intricate features to classify images accurately. Its application facilitates early diagnosis of eye conditions, enabling timely medical intervention and potentially improving patient prognosis.
XCEPTION	The Xception model utilizes a highly efficient convolutional neural network architecture trained on annotated retinal or fundus images. With its advanced feature extraction capabilities, it accurately classifies diverse eye diseases, enabling early diagnosis and timely medical intervention.

INCEPTION V3	The Inception v3 model harnesses a sophisticated convolutional neural network architecture tailored for image classification tasks. Trained on annotated retinal or fundus images, it adeptly discerns nuanced features to classify diverse eye diseases accurately.
---------------------	--

4.2 Initial Model Training, Model Validation and Evaluation

Model Training:

CNN :

```
# Define the CNN model architecture
def create_cnn_model(input_shape, num_classes):
    model = models.Sequential()

    # Add convolutional layers
    model.add(layers.Conv2D(32, (3, 3), activation='relu',
input_shape=input_shape))
    model.add(layers.MaxPooling2D((2, 2)))
    model.add(layers.Conv2D(64, (3, 3), activation='relu'))
    model.add(layers.MaxPooling2D((2, 2)))
    model.add(layers.Conv2D(64, (3, 3), activation='relu'))

    # Flatten the output and add dense layers
    model.add(layers.Flatten())
    model.add(layers.Dense(64,
activation='relu', kernel_initializer='random_uniform'))
    model.add(layers.Dense(num_classes, activation='softmax',)) # Output layer

    model.compile(loss='categorical_crossentropy',
                  optimizer=RMSprop(learning_rate=0.001),
                  metrics=['accuracy'])

    return model

input_shape = (64, 64, 3)
num_classes = 4

cnn_model = create_cnn_model(input_shape, num_classes)
```

```
eye_detection = cnn_model.fit_generator(  
    x_train,                          # Training data generator  
    steps_per_epoch=len(x_train),     # Number of batches  
    epochs=80,                        # Number of epochs  
    validation_data=x_test,  
    validation_steps=len(x_test)      # Number of batches to yield from the  
validation generator per epoch  
)
```

VGG 19:

```
vgg = VGG19(include_top=False, input_shape=(64, 64, 3), weights='imagenet')  
for layer in vgg.layers:  
    layer.trainable=False  
x = Flatten()(vgg.output)  
output = Dense(4, activation='softmax')(x)  
vgg19 = Model(vgg.input, output)
```

```
from tensorflow.keras.optimizers import RMSprop  
optimizer = RMSprop(learning_rate=0.001)  
vgg19.compile(loss='categorical_crossentropy',  
              optimizer=optimizer,  
              metrics=['accuracy'])
```

```
vgg_detection=vgg19.fit(x_train, validation_data=x_test, epochs=50, steps_per_epoch=len(x_train), validation_steps=len(x_test))
```

Inception V3:

```
# Load pre-trained Inception model  
base_model = InceptionV3(weights='imagenet', include_top=False,  
input_shape=input_shape)
```

```
# Freeze the pre-trained layers  
for layer in base_model.layers:  
    layer.trainable = False
```

```
x = GlobalAveragePooling2D()(base_model.output)  
x = Dense(256, activation='relu')(x)
```



```
predictions = Dense(num_classes, activation='softmax')(x)
```

```
from tensorflow.keras.callbacks import EarlyStopping
# Define early stopping callback
early_stopping = EarlyStopping(monitor='val_accuracy', patience=3,
restore_best_weights=True)
```

```
detection_inception = model.fit(train, steps_per_epoch=50, epochs=epochs,
validation_data=test, validation_steps=len(test))
```

Xception:

```
base_model = Xception(weights='imagenet', include_top=False,
input_shape=input_shape)
```

```
# Freeze the pre-trained layers
for layer in base_model.layers:
    layer.trainable = False
```

```
x = GlobalAveragePooling2D()(base_model.output)
x = Dense(256, activation='relu')(x)
predictions = Dense(num_classes, activation='softmax')(x)
```

```
model = Model(inputs=base_model.input, outputs=predictions)
```

```
# Compile the model
model.compile(optimizer=Adam(learning_rate=0.001),
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

```
# Train the model
xception_model = model.fit(
    x_train,
    steps_per_epoch=50,
    epochs=epochs,
    validation_data=x_test,
    validation_steps=len(x_test)
)
```

Model Evaluation:

CNN:

```
results = cnn_model.evaluate(x_test)
```

VGG 19:

```
results_vgg=vgg19.evaluate(x_test)
```

Inception V3:

```
# Evaluate the model on test data
test_loss, test_accuracy = model.evaluate(test, steps=len(test))
print("\nTest Loss:", test_loss)
print("Test Accuracy:", test_accuracy)
```

Xception:

```
# Evaluate the model on test data
test_loss, test_accuracy = model.evaluate(x_test, steps=len(x_test))
print("Test Loss:", test_loss)
print("Test Accuracy:", test_accuracy)
```

5. Model Optimization and Tuning Phase

5.1. Tuning Documentation

Model	Tuned Hyperparameters
CNN	<pre>model.compile(loss='categorical_crossentropy', optimizer=RMSprop(learning_rate=0.001), metrics=['accuracy']) return model</pre> <p>Loss Function: <code>loss='categorical_crossentropy'</code> - Determines the error between predicted and actual output for multi-class classification tasks.</p> <p>Optimizer: <code>optimizer=RMSprop(learning_rate=0.001)</code> - RMSprop optimizer with a learning rate of 0.001, adjusting the step size during training for better convergence.</p> <p>Metrics: <code>metrics=['accuracy']</code> - Evaluation metric to measure the proportion of correctly classified examples out of the total during training and testing.</p> <pre>detection=model.fit_generator(x_train,steps_per_epoch=46, epochs=100,validation_data=x_test, validation_steps=20)</pre> <p>After applying 100 epochs we got 76% accuracy so we decided to reduce the epoch</p>

	<pre data-bbox="446 262 1388 357">] detection2=cnn_model.fit_generator(x_train,steps_per_epoch=50, epochs=80,validation_data=x_test, validation_steps=20)</pre> <p data-bbox="1404 357 1477 388">After</p> <p data-bbox="438 420 1477 514">applying 80 epochs we got 85% of accuracy so we thought of increasing epoch to increase the accuracy.</p> <pre data-bbox="446 609 1388 703">) detection1=model.fit_generator(x_train,steps_per_epoch=46, epochs=70,validation_data=x_test, validation_steps=20)</pre> <p data-bbox="438 787 1477 892">But after applying 70 epochs we got 82% of accuracy so we decided to go with 80 epoch only.</p>
--	--

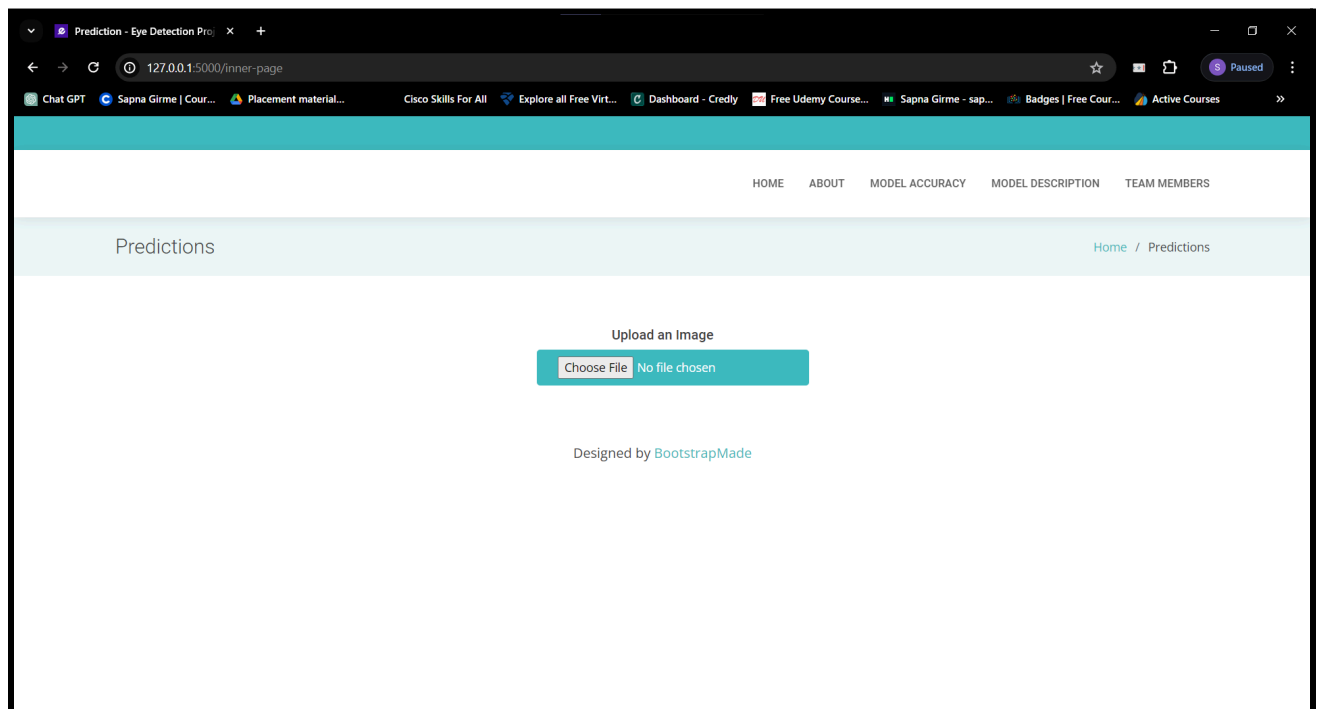
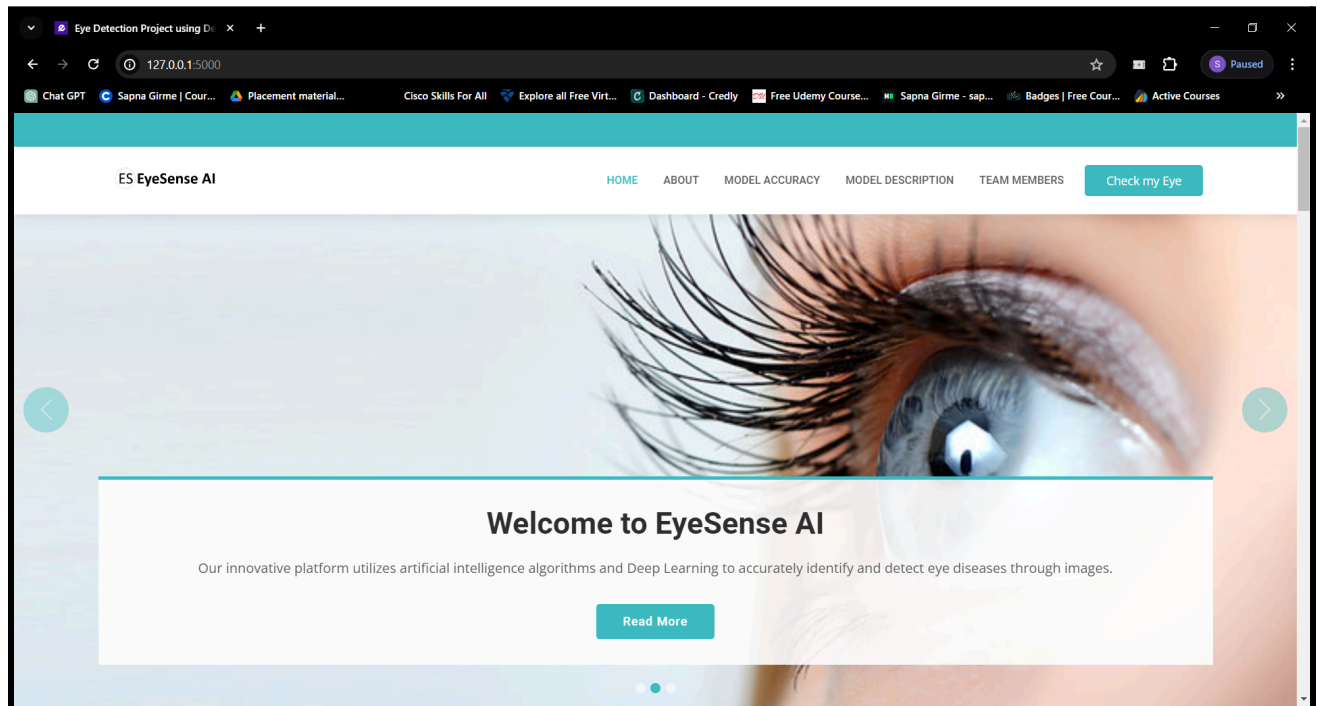
5.2. Final Model Selection Justification

The decision to select a Convolutional Neural Network (CNN) as the final optimized model for eye disease detection was based on its robust capability to extract relevant features from raw image data and its accuracy around 85%. We have built VGG19, XCEPTION, INCEPTION V3 and CNN out of which CNN had the highest accuracy.

To refine its performance and ensure robustness, the CNN model underwent further optimization through techniques such as hyperparameter tuning, data augmentation methods. These enhancements were crucial in enhancing the model's accuracy, reliability, and generalizability for accurately diagnosing eye diseases from image data.

6. Results

6.1. Output Screenshots



Prediction - Eye Detection Proj x +

127.0.0.1:5000/inner-page


Chat GPT Sapna Girme | Cour... Placement material... Cisco Skills For All Explore all Free Virt... Dashboard - Credly Free Udem Course... Sapna Girme - sap... Badges | Free Cour... Active Courses

HOME ABOUT MODEL ACCURACY MODEL DESCRIPTION TEAM MEMBERS

Predictions [Home / Predictions](#)

Upload an Image

Choose File Cataract.jpg



Click here to see result

Result: You have cataract type of eye disease.

Prediction - Eye Detection Proj x +

127.0.0.1:5000/inner-page


Chat GPT Sapna Girme | Cour... Placement material... Cisco Skills For All Explore all Free Virt... Dashboard - Credly Free Udem Course... Sapna Girme - sap... Badges | Free Cour... Active Courses

HOME ABOUT MODEL ACCURACY MODEL DESCRIPTION TEAM MEMBERS

Predictions [Home / Predictions](#)

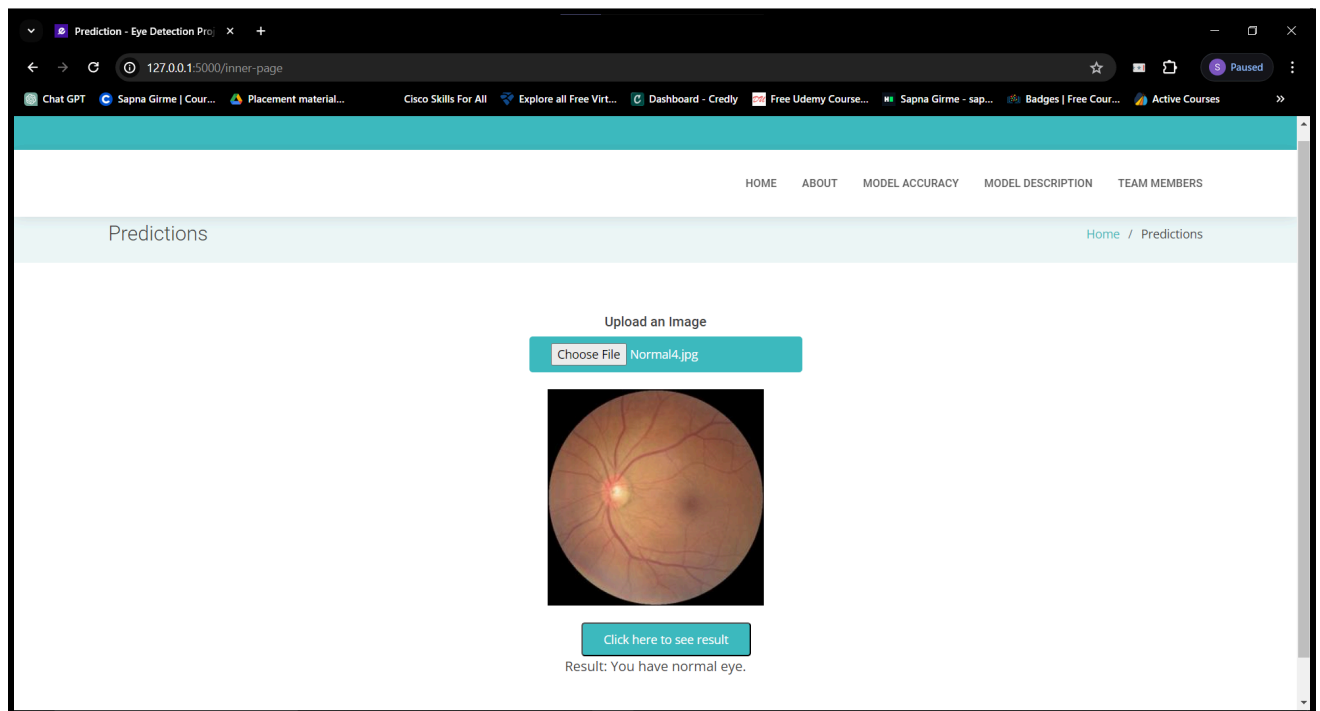
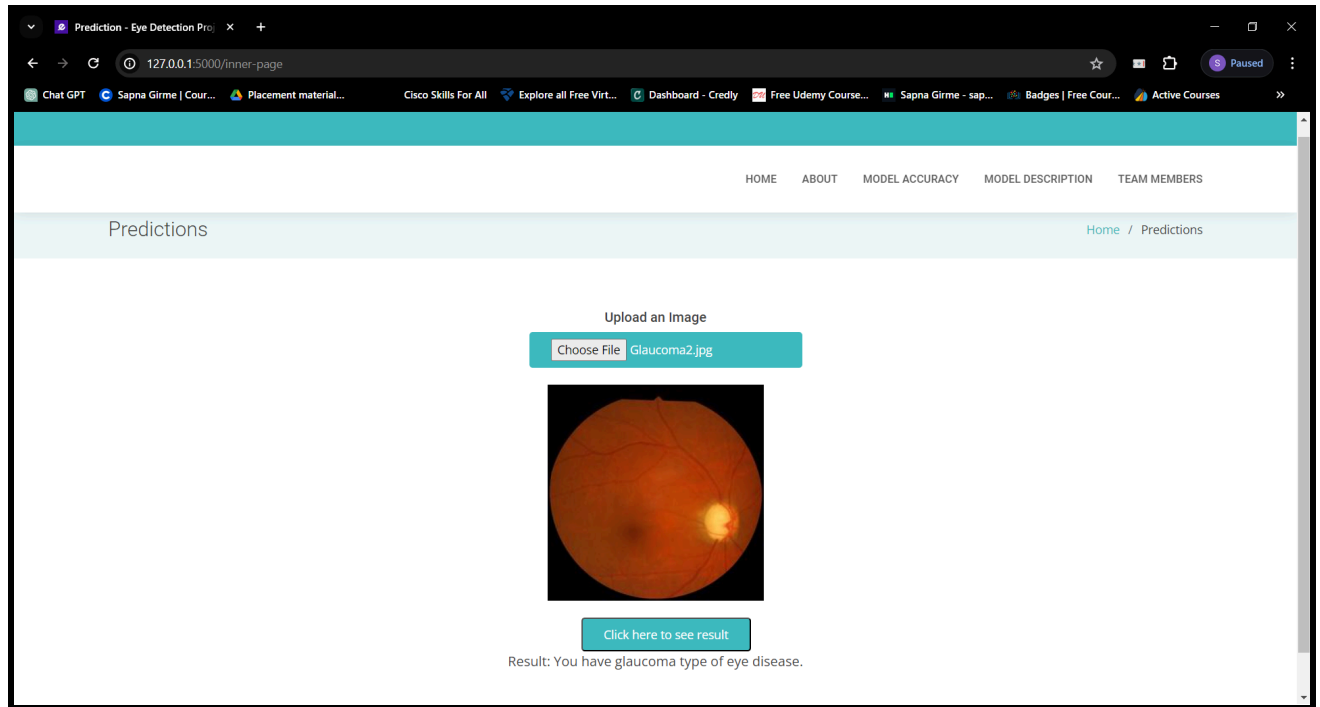
Upload an Image

Choose File Diabetic_Retinopathy.jpeg



Click here to see result

Result: You have diabetic retinopathy type of eye disease.



7. Advantages & Disadvantages

Advantages:

Using deep learning for eye disease detection offers significant advantages including high accuracy due to its ability to identify subtle patterns, early detection capabilities facilitating timely treatment interventions, automation of screening processes enhancing efficiency and scalability, rapid analysis of medical images enabling quick diagnoses especially in emergency scenarios, potential for personalized treatment plans based on individual patient data, cost-effectiveness by reducing late-stage treatment expenses, and continuous improvement through ongoing training on vast datasets, collectively promising improved patient outcomes, streamlined healthcare workflows, and optimized resource allocation within healthcare systems.

Disadvantages:

While deep learning offers high accuracy and automation in eye disease detection, challenges include the need for large, labeled datasets, potential lack of transparency in decision-making, risks of overfitting and biases, and the complexity of deployment in clinical settings, which may hinder widespread adoption.

8. Conclusion

In conclusion, the development of an automated eye disease detection system using deep learning techniques represents a significant advancement in the field of medical image analysis. Throughout this project, we successfully applied a diverse dataset sourced from Kaggle, comprising images categorized into cataract, glaucoma, diabetic retinopathy, and normal eye conditions.

Multiple deep learning models, such as Convolutional Neural Networks (CNN), XCEPTION, Inception V3, and VGG19, were built and evaluated for their ability to accurately classify eye diseases. The CNN model performed the best, achieving an impressive accuracy rate of 85% on the validation set, and was subsequently deployed in a web application using Flask for real-time disease prediction.

The web application provides a user-friendly interface where healthcare professionals can upload eye images and receive instant diagnostic results, facilitating early detection and timely intervention. This project underscores the potential of deep learning in revolutionizing healthcare by offering automated diagnostic tools that augment the capabilities of medical practitioners.

9. Future Scope

Expand the scope of the automated eye disease detection system by adding disease categories beyond cataract, glaucoma, diabetic retinopathy, and normal. This could include other common eye conditions such as macular degeneration, retinal detachment, or optic neuritis, etc.

Implement continuous learning techniques so that the model can adapt and improve over time. Utilize methodology like transfer learning, online learning, or active learning to update the model with new labeled data and advancements in the field of ophthalmology.

Develop capabilities for patient-specific risk assessment and personalized disease management recommendations based on the automated diagnostic results. Incorporate patient demographics, medical history, and lifestyle aspects to deliver personalized insights and treatment recommendations to improve individual patient care.

10. Appendix

10.1. Source Code

Source Code Link

Training File:

 Model Training.ipynb

Testing File:

 Model Testing File.ipynb

10.2. Project Demo Link

Project Demo Link:

 Project Video Demonstration .mp4